J Intell Robot Syst (2017) 85:145–165
DOI 10.1007/s10846-016-0370-z

Emergency Flight Planning for an Energy-Constrained Multicopter

Alec J. Ten Harmsel · Isaac J. Olson · Ella M. Atkins

Received: 22 December 2014 / Accepted: 11 April 2016 / Published online: 10 May 2016

Abstract Small Unmanned Aircraft Systems (UAS) have diverse commercial applications. Risk mitigation techniques must be developed to minimize the probability of harm to persons and property in the vicinity of the aircraft. This paper presents an emergency flight planner combining sensor-based and map-based elements to collectively plan a landing path for a UAS that experiences an unexpected low energy condition while flying over a populated area. Focus is placed in this work on the use of public databases of population distribution, structure locations, and terrain to create an efficient-to-access cost map of the data. Safe landing plans are generated with an A* search algorithm shown to be feasible for real-time use with the cost map. Simulation-based case studies are presented of a quadrotor UAS operating within New York City to illustrate how different cost terms impact optimal path characteristics.

Keywords Path planning · Cost modeling · Risk mitigation

A. J. Ten Harmsel (✉) · I. J. Olson · E. M. Atkins
Department of Aerospace Engineering, University of Michigan, 1221 Beal Avenue, Ann Arbor, MI 48109, USA
e-mail: talec@umich.edu

Ella M. Atkins
e-mail: ematkins@umich.edu

1 Introduction

Small Unmanned Aircraft Systems (UAS) are expected to experience a period of significant growth as regulatory policy supports them in the coming years [2, 12, 19, 25]. Many applications such as infrastructure inspection, law enforcement, and package delivery all require the UAS to fly over and within populated areas.

A major challenge for urban UAS flight is to reduce the risk they pose to surrounding persons and property. In the case of an in-flight failure [17], it will be of the utmost importance that the UAS be capable of determining and performing actions to reduce or eliminate the risk it poses to its surroundings. Additionally, the vehicle must be able to make emergency management decisions onboard as loss of communication link is generally recognized as one of the most common failures.

Another commonly-experienced failure is running low on fuel or battery energy. If an unexpected low energy condition occurs in an urban area, the UAS must ensure it does not introduce unacceptable risk to people or property when it lands. In these areas, the UAS may be flying directly above people or property, and may therefore need to maneuver toward a new location that poses minimum risk to people and property before it lands, including flying a path in the air that minimizes risk in case of total failure leading to a crash.

A meta-level emergency landing planner, shown in Fig. 1, is proposed to calculate a safe path for a small UAS that senses unexpectedly low energy reserves while flying over or within a populated urban environment. Two emergency planning algorithms, a sensor-based planner and a map-based planner, are combined to maximize information used in onboard decision-making. The sensor-based planner determines the safest visible touchdown site and final approach route. The map-based planner uses a pre-computed cost map to determine the safest transit to a landing site area that may be beyond sensor range or visibility. The current safety of the sensed environment around the UAS as well as its energy level determine which of the two planners to use; the map-based planner is only used if the immediate environment is unsafe and energy levels are high enough to travel to a new, safer landing location. Once at the new, safer location, the safety of the environment and energy levels are again used to determine whether to land immediately with the sensor-based planner or run the map-based planner again. This process repeats as needed until a successful landing is achieved, resorting to sensor-based landing in the event of imminent motor shutdown. Landing due to imminent motor shutdown may occur in an unsafe area. In this case, the sensor-based planner, using one of a variety of techniques [1, 11, 23, 24], is responsible for executing the lowest-risk landing plan.

The primary contribution of this paper beyond the architecture summarized above is in map-based planning. Specifically, this paper proposes a novel method for processing online databases and fusing their data into a single cost function appropriate for flight planning. The map-based planner is separated into two major modules; an offline cost map generation module and an online planner. The cost map is generated using publicly-accessible databases of population, structure, and terrain data. Weighting factors are used to merge population, structure, and terrain data to minimize risk

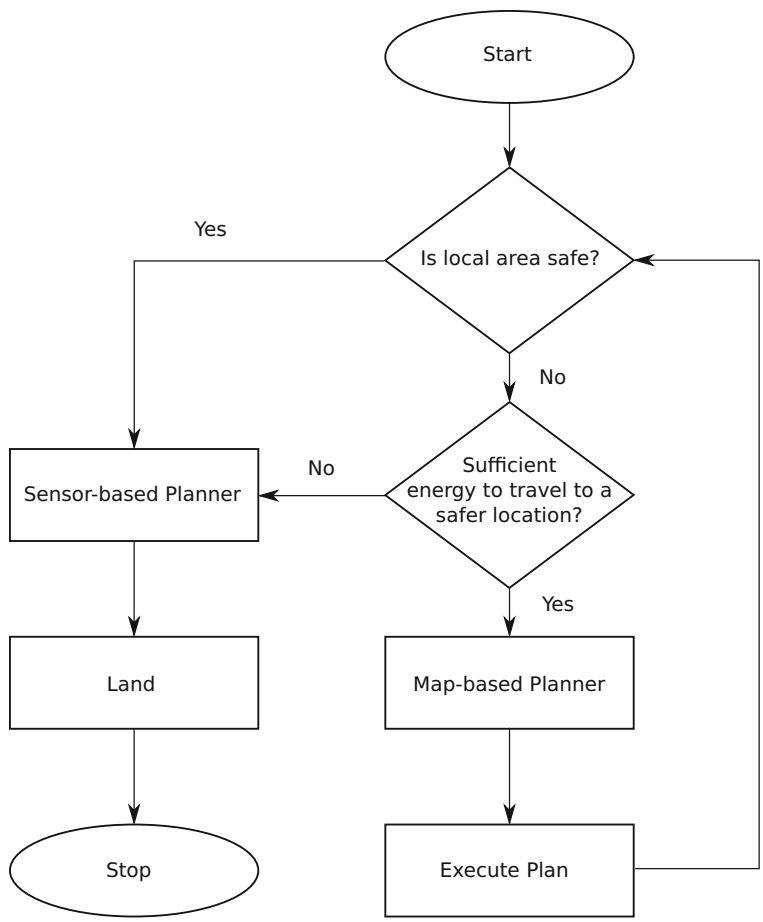


Fig. 1 Emergency landing planning architecture. Local area landing viability and energy level remaining for transit are preconditions to the use of the map-based emergency flight planner

to people, property, and the UAS. From this data, a cost map generation module chooses optimal landing sites offline, based on the assigned cost of locations on the ground, as well as some key building rooftops. Calculating the cost map offline and loading it onto the UAS is the key to managing planning complexity, as merging these datasets requires more computing power than is available on current small UAS. The planner module loads the cost map and associated pre-processed landing sites, and uses the well-known A* algorithm to build optimal landing paths. Using an estimate of the available flight range from remaining battery capacity, the planner bounds the search space, reducing the run time. Additionally, the use of a heuristic based on straight line distance reduces the amount of the search space that is actually searched by the planner. The effects of using a heuristic-guided search, pre-computing the cost map and landing sites, and writing the planner in a compiled language allow the planner to be run in real-time on a small UAS.

Performance of the emergency landing planner is evaluated in simulation by obtaining population, building, and terrain maps for the midtown Manhattan region of New York City. A series of case studies illustrate how the map-based planner can provide a safe landing capability for a multicopter UAS flying in and over one of the most densely populated cities in the United States (US).

This paper is organized as follows: Section 2 reviews related work; Section 3 gives an overview of the vehicle model; Section 4 describes the environment model; Section 5 describes planner cost and constraint terms; Section 6 details the flight path planning algorithm; Section 7 presents case studies for a quadrotor operating in Manhattan; Section 8 gives conclusions and considerations for future work.

2 Background

Emergency flight planning has been studied for manned and unmanned aircraft. Model and data-based planners build landing plans from stored information, while sensor-based planners reactively construct flight plans as the environment is observed. Both have proven beneficial. Model-based flight planning, for example, can enable an aircraft that has lost thrust to glide to a known runway [3, 15] or guide a rotorcraft to a terminal approach waypoint where a vision-based system can autonomously land on a marked target [21, 22]. Researchers have also worked to classify landing sites based on data acquired during flight [11]. Such data could be used in real-time should an aircraft be able to survey a large area prior to landing, or it could be collected into a large database suitable for map-based planning as is proposed in this work.

Vision-based landing systems have also been proven effective as sensor-based landing planners, even in situations with wind and a moving landing platform [18]. Optic flow has been proven valuable as a means of terminal approach guidance [23], while visual SLAM (simultaneous localization and mapping) provides a means to avoid enroute obstacles even at higher flight speeds [1]. All this work is complementary to the emergency landing planner proposed in this paper. Manned fixed-wing emergency landing planners provide motivation for beyond-the-horizon planning even when near-term landing is required. Sensor-based guidance is an essential follow-on to model-based planning to allow a safe terminal approach and landing once the UAV is within sensor, e.g., visual, range of the landing area. Techniques such as visual SLAM are also essential in case dynamic or unmapped obstacles are encountered during transit to the selected landing site.

Path planning through an environment with obstacles is a widely-studied field divided into two basic categories: search (optimization) using stored models and data, and reactive planning that updates plans in response to incoming data. Optimization-based approaches tend to be more computationally-intensive as they must compare potentially numerous solution paths to identify a minimum-cost plan. Alternatively, reactive planning approaches are difficult to apply in situations where resources do not allow a vehicle to "turn back" should an over-the-horizon region prove difficult or unsafe to navigate or use for landing. As discussed above, we believe a judicious combination of these two basic approaches is best: apply an optimal search approach to guide the aircraft to a site expected to be safe, then apply a sensor-based planning method for safe terminal approach and landing. Substantial previous work has focused on sensor-based emergency landing planning solutions. Reference [6] describes sensor-based planning using rapidly exploring random tree (RRT)* [13] and anytime A* approaches, with plans quickly retrieved from a pre-computed ensemble. This work is similar in that

preprocessing is key to fast planner response but distinct in that [13] defines plans based on local environment sensing rather than map databases. Choudhury et al. [7] propose RRT* for to define a family of alternate emergency landing plans from which the pilot ultimately selects. As will be described below, our contributions are focused on database fusion into a multi-objective cost map appropriate for urban area emergency flight planning, and in assessing application of map-based planning to a Manhattan case study. We utilize a straightforward A* approach to optimally search and assess a distance heuristic, a sufficient strategy that leaves research in establishing specific tradeoffs between different search strategies for future work.

3 Vehicle Model

The vehicle model is based on a quadrotor UAS. It is a combination of a curve-fit energy equation based on discharge curves of lithium polymer batteries and experimental data taken from the Michigan Autonomous Aerial Vehicles (MAAV) team, a UAS student team at the University of Michigan[1], and a simple equation validated with benchtop testing that translates motor force into approximate current draw from the onboard battery. Motor force can be approximately related to current draw as all forces except the force required to lift the vehicle are negligibly small [4]. This model allows the path planner to consider energy efficiency while planning without adding the significant computational complexity during the search imposed by optimizing maneuvers/accelerations.

3.1 Physical Model

To model quadrotor instantaneous power requirements, total motor force and in turn current must be approximated. The quadrotor UAS velocity model used in this work is straightforward with total required force output from the motors computed as the sum of force to overcome gravity g and to provide the desired acceleration or change in velocity dv, the difference between the desired velocity v_{next} and the current velocity v over a normalized time period. The simulated quadrotor travels through each cell in the discretized cost matrix (described in Section 4) with a constant velocity and transitions to the next velocity when moving to the next cell. The required force from the motors is a function of dv and g as shown below.

$$dv = v_{next} - v \tag{1}$$

$$f = \text{norm}\left((dv + g) \cdot m\right) \tag{2}$$

This physical model is used as an input to the energy model, which in turn is an input to the path planner. For different vehicle configurations, different models may be needed to accurately model energy requirements. In the case of hybrid air vehicles that can takeoff and land vertically but also aerodynamically lift themselves during forward flight, the flight planning energy model must also account for the costs associated with transitioning between forward and vertical flight configurations.

3.2 Energy Model

It is critical to properly model remaining battery capacity at each step in the landing path. As seen in Fig. 2, lithium polymer batteries maintain a fairly constant voltage for the majority of the discharge period followed by a quick drop-off.

Change in battery energy dB is computed from f defined in Eq. 2. A function $D(f)$ provides a simple translation of force to expected current draw:

$$dB = D\left(f\right) \tag{3}$$

Current draw function D was developed from curve fitting experimental data collected during motor tests for the MAAV quadrotor which uses Axi Gold 2212 motors, Castle Creations Pheonix 25 electronic speed controllers and 9 inch propellers with 3 blades. Tests were conducted when operating at the nominal 3.4 V per cell. This curve is shown in Fig. 3. To incorporate energy costs for fixed-wing or hybrid air vehicle configurations, a different current draw function D would be substituted into the planner.

A low-energy condition requires generation of a trigger for the planner. Such a trigger could be a simple voltage threshold. However, a proper trigger would reflect current draw over the remaining flight profile which are in turn a function of expected environmental conditions (e.g., temperature, wind). If a

[1]The first author is a member of the MAAV team and the other authors are advisors to the MAAV team.

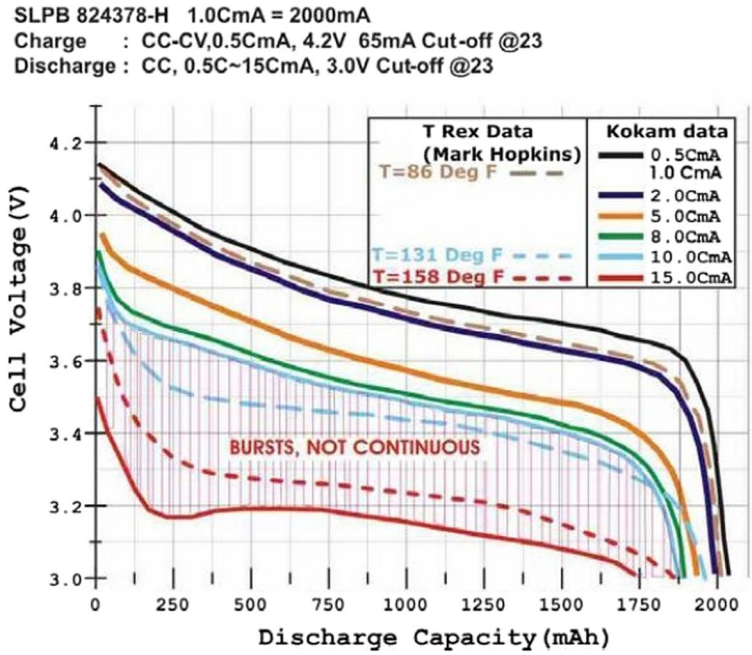


Fig. 2 Lithium Polymer Battery Discharge Curves. Note that discharge ratings are in terms of the capacity of the battery 'C' so a battery that has a capacity of 2Ah and discharged at a rate of 1C will be outputting 2A. (http://www.droidforums.net/forum/attachments/smartphone-battery-discussion/50553d1338473331-degradation-battery-discharge_curves_1.jpg)

mission is entirely pre-planned, expected energy levels can be predicted based on current environment observations. However, the problem of correctly and consistently generating the low-energy trigger itself requires a careful algorithm that incorporates voltage measurements and power outputs relative to those expected. The work presented in the paper focuses on landing planning given a triggered low energy state; details in robust low energy triggering are therefore not considered further.

4 Environment Model

The environment is modeled by a three dimensional discretized cost matrix. This matrix is populated by

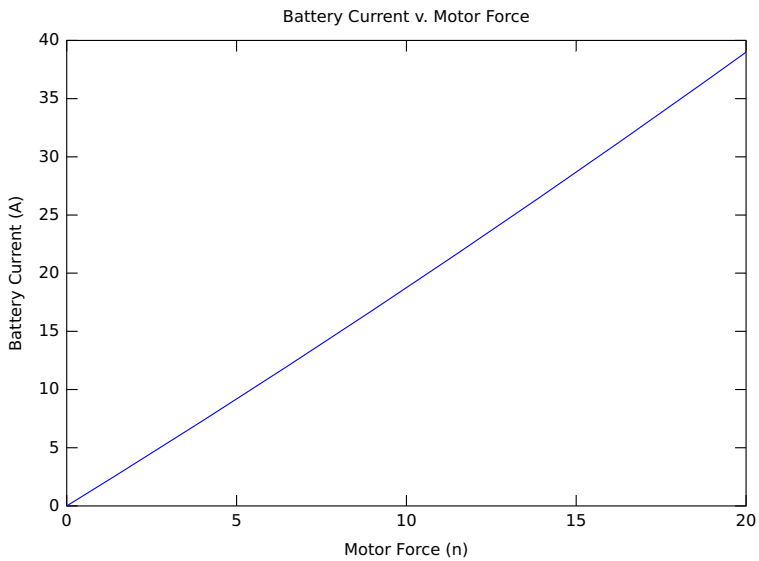


Fig. 3 Current drawn by the battery as a function of total motor force required by the vehicle

processing multiple databases offline, combining the resulting data, and transforming the data into the cost matrix. After the data transformations described below have been performed, the cost can be calculated as discussed in Section 5.

While a cost matrix may contain data from any number of sources as long as each data source is mapped to a cost function, our case study focuses on three main datasets: population density, structural data, and terrain information. These datasets are available from multiple government agencies, and all data used are released under open licenses permitting a variety of uses. Satellite imagery of a subsection of Manhattan, the specific area of New York City (NYC) used in our simulations, is shown in Fig. 4.

To supplement information loaded into the UAS before flight, real-time connection to the cloud may also be possible. If this link is reliable, the cloud could utilize higher-accuracy vehicle and environment models to assist with emergency flight planning up to the limits of available link bandwidth and cloud processing availability. However, any connection to the cloud may be unreliable as the vehicle traverses the urban environment, and the cloud may not guarantee meeting real-time planning constraints. In any case, the cloud may be used to deliver updates to onboard maps, models, and software while the vehicle is not flying.

4.1 Databases

Data for the population density of New York City and the surrounding area were acquired from the 2010 US Census, gathered and maintained by the US Census Bureau. The dataset contains fields describing the population density and the boundaries of the census blocks. Census blocks are irregularly shaped and do not all cover the same amount of geographical area, but complete coverage of the Manhattan area is provided. The US Census Bureau does not provide uncertainty estimates [5, ch. 7].

Fig. 4 Satellite image of the area of Manhattan used in case studies for the path planner (Google maps)

The NYC Department of City Planning (NYC-DCP) has released the Primary Land Use Tax-lot Output (PLUTO) dataset, which contains information about every taxable lot in NYC. We incorporate PLUTO building data into our simulator that can be easily extended or updated as needed. The following PLUTO dataset fields are used in this model:

- Number of Floors
- Lot Bounding Box
- Lot Boundary Vector Array

This dataset is contained in an ESRI Shapefile [10]. The full PLUTO dataset is separated by region, and the Manhattan region dataset is used in our simulation. The Shapefile comes with location information (e.g. the Bounding Box and Vector Array) listed in New York State Plane coordinates. The NYC-DCP also does not provide uncertainty information with its dataset. The Shapefile is converted to latitude/longitude coordinates for use in our algorithms by the open-source Geographic Information System software QGIS.

Elevation and terrain class data are available from the US Geological Survey (USGS). The 2006 National Land Cover Database provides information about the type of terrain for the entire US. The 2009 National Elevation Dataset provides information about the slope of the terrain for the entire US. Neither dataset provides uncertainty information, further supporting the choice of a deterministic planner that does not bias solutions based on site-specific data uncertainty estimates.

4.2 Map Generation

We transform individual data into multiple intermediate maps that represent the environment in an intuitive manner. Population data is transformed into a population density map, the structural data into a building height map, and the terrain data into both a map of terrain slopes and terrain types.

4.2.1 Population Map Generation

Using published latitude, longitude, population, and area values from the population density dataset, a population density map was created using discrete data points and interpolating a smooth surface between them. As mentioned above, census blocks are rarely simple shapes that can be fit to a grid; this interpolated representation provides an accurate approximation of the data.

The population density dataset is strictly based on the permanent residences of people in New York; no data on population density at various times during the day is available, hence there is no way to know using the population density map how populated a particular area, e.g., Central Park or Times Square, will be at a given point in time. Naturally, a deployed UAS should ultimately supplement database information with real-time sensor data (e.g., cameras) to ensure it can account for unexpected people and objects near the emergency landing site. If a connection to the cloud is available, a UAS could potentially use cell phone data to compute a live population density map [9].

4.2.2 Structural Map Generation

Using the fields enumerated above, a height map of a subsection of Manhattan was created. Initially, the vector boundary information was read from the Shapefile. The results of reading the vector information can be seen in Fig. 5.

This vector boundary data is further transformed by adding height data also available in the Shapefile. The only height data available in the Shapefile is the number of floors, so a height per floor is assumed to be 4m. This height map is shown in Fig. 6. The structural map is populated solely with building data, producing a map with some uncertainty, but no data on obstacles other than buildings is available.

4.2.3 Terrain Map Generation

Elevation data from the terrain dataset was differentiated into a slope map for Manhattan. This slope map can be used to determine better landing locations for the vehicle, as areas of higher slope may not support a safe landing. Of course, Manhattan is a generally flat area, and, as such, the slope map does not add much to our simulations. However, in any areas with rugged terrain, knowledge of terrain slopes might be of significant influence in selecting an optimal landing location.

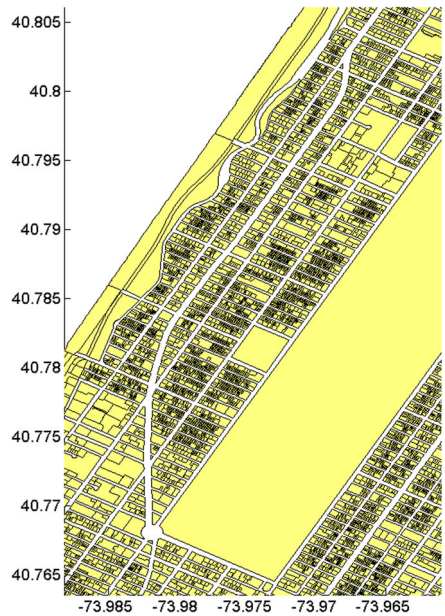


Fig. 5 Matlab representation of the Shapefile, showing all boundaries of all lots within the area of Manhattan used in our simulation

Choosing the best landing location also requires knowledge of the terrain type; landing on soft grass may be better for the safety of the UAS than landing on pavement. The terrain class data from the USGS gives information on what type of terrain exists at a given latitude/longitude location. Some example terrain classes are water, open development, low development, barren, forested, high development, and wetlands. There are 13 classes in all. A terrain types map is created from this data, where each type is represented by a distinct integer.

5 Costs and Constraints

The map-based planner is optimal and constrained. Constraints and costs are discussed below. The UAS planner takes two cost terms - the movement cost C_M and the environment cost C_E - and sums them to provide the total cost function C, as shown in Eq. 4.

$$C = C_M + C_E \tag{4}$$

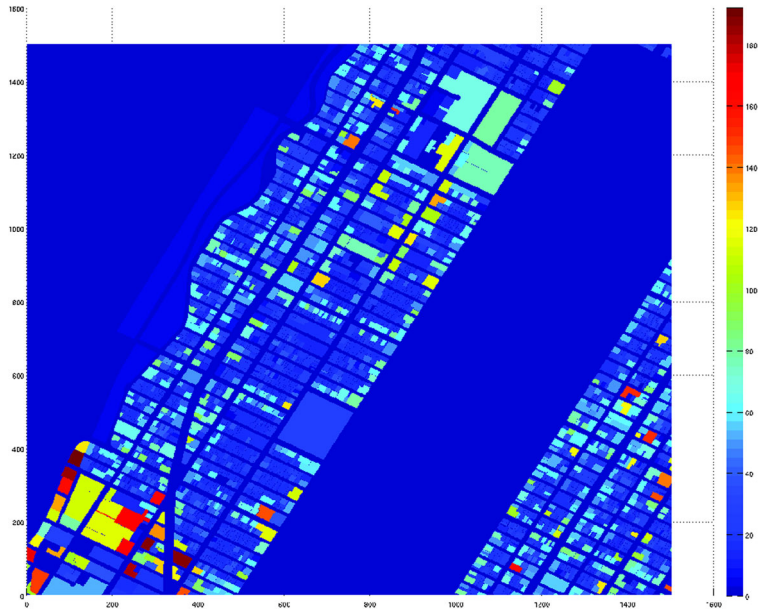


Fig. 6 Height map created from the Shapefile. Dark blue represents no buildings; light blue to yellow to red depict increasing building heights

5.1 Constraints

There are multiple constraints our planner must satisfy to plan safe landing paths. The planner must not generate a path that requires more than the available stored energy. Additionally, the planner must not plan a path that goes through a building. In this work the planner is also constrained to never generate a path over very high population areas or areas with exceptionally bad terrain.

To simplify constraint processing in this work, environmental constraints are represented by a cost of 1.0 in each map cell. As will be described further below, the A* planner will not explore cells with a cost of 1.0. The constraint on population density is defined as a population density higher than one person inside of the UAS safety radius r. This safety radius is the distance that a person should be away from a vehicle while landing in order to be safe. The safety radius does not relate to the resolution of the planner; it solely affects map generation. For our simulated UAS, we chose 2m as our safety radius based on experience with conducting over 1000 logged indoor flight tests with the MAAV quadrotor UAS. The MAAV vehicle has a maximum in-flight radius of 0.3m; since a multi-rotor is capable of vertical landing, a 2m safety radius provides the necessary clearance. This results in a maximum population density of 0.0795 persons/m^2, as shown in Eq. 5.

$$\frac{1}{\pi r^2} = \frac{1}{4\pi} = 0.0795 \text{ persons}/m^2 \tag{5}$$

The terrain constraint works analogously to the population density constraint. If a UAS was flying in an area with mountainous or otherwise potentially dangerous terrain, types representing dangerous terrain would correspond with a cost of 1.0. Specifics of terrain costs are not discussed further here as terrain costs for the urban environment of Manhattan are dominated by man-made structure costs and constraints, and structure data is readily available.

5.2 Movement Costs

The simple model of UAS movement costs was described in Section 3, and the equation for finding the required battery current from the physical maneuvers of the UAS is provided in Eq. 3. The equation for dB as well as the amount of energy available, B_0, where the low energy condition is triggered for this work, together define Eq. 6, a cost function for total battery energy drawn over a single maneuver. A maneuver is

a change from velocity v to v_{next} occurring over a normalized time interval.

$$C_A(v, v_{next}) = dB/B_0 \tag{6}$$

The acceleration costs are then used to create the overall path, or movement, cost C_M by summing along a potential path to be taken by the vehicle. This calculation considers a potential path or landing flight plan P with number of cells n, finds the accelerations necessary to move to each location, and generates C_M as a sum of acceleration costs as shown in Eq. 7.

$$C_M(P) = \sum_{i=2}^{n-1} C_A(P_i - P_{i-1}, P_{i+1} - P_i) \tag{7}$$

C_M represents the fraction of the currently available energy that would be used during the simulated path traversal. A cost of 1.0 or greater represents a path that could not be traversed; if no solution is identified by the map-based planner that satisfies this constraint the sensor-based planner would be activated to land immediately.

5.3 Environment Costs

Minimizing damage, first to people, second to property, and third to the UAS, is the objective of the environment cost function C_E. There are three sub-functions that are weighted and summed to form the environment cost: C_p, the population cost to protect people; C_s, the property repulsion cost to protect physical property; and C_t, the terrain cost to protect the UAS. Each of these sub-functions transforms their respective two dimensional maps into three dimensional cost maps.

Depending on the environment and needs of the UAS, the weighting factors for each particular cost may be adjusted as necessary. Note that since all terms are normalized the weighting terms will directly indicate the relative values of the different cost terms. The environment cost at location L is defined as:

$$C_E(L) = \begin{bmatrix} w_p & w_t & w_s \end{bmatrix} \begin{bmatrix} C_p(L) \\ C_t(L) \\ C_s(L) \end{bmatrix} \tag{8}$$

The three dimensional population cost map C_p was created by normalizing the population density map with the result of Eq. 5, 0.0795. As the population density map is two dimensional, population cost is assumed the same at all altitudes. A 2-D population cost map is shown in Fig. 7.

The three dimensional structure cost map C_s was created by using the height map to create a three dimensional occupancy grid [8] and adding repulsion cost around the structures, which inflates the cost in the cells surrounding structure-occupied cells. Because the cells near structures have a higher cost than those slightly farther away, the vehicle will tend to keep a wider safety margin from structures while maintaining a reasonably short path length.

To create the repulsion cost, cost term C_R is applied to each cell in the occupancy grid. C_R searches the space around the current cell c, calculating a distance-based decaying cost for each cell in the search space, and applies the maximum of these distance-based costs to the current cell c. Repulsion cost decreases linearly with increase in distance, using a vector s of cells in the search space. The formula for the repulsion cost function C_R is given by:

$$w = \max(\text{norm}(s - c)) \tag{9}$$

$$C_R(c) = \max\left(1 - \frac{C_s(s) \cdot \text{norm}(s - c)}{w}\right) \tag{10}$$

The number of cells in s can be adjusted, as s is computed by defining a repulsion field width and adding all cells within this width as defined by c to s. C_R is applied to every cell in the structural occupancy grid [8], resulting in the final structural cost map C_s. An example two dimensional repulsion field is shown in Fig. 8. The actual repulsion fields generated in our case study are three dimensional, creating regions of higher cost both along the sides of known structures as well as above them.

The three dimensional terrain cost map C_t was created by combining two different terrain sub-cost maps and extending their sums into three dimensions in the same manner as the population cost map.

The first terrain sub-cost term, the terrain slope cost map, was generated by normalizing the slope map to 45°, a large value for Manhattan but applicable for most terrain given that a quadrotor could not land on a surface with greater slope. The second terrain sub-cost term, the terrain class cost map, transforms different types of terrain into costs. Water was given higher cost than land, and more developed land has higher cost than less developed land. The costs for each type were assigned arbitrarily but intelligently. Water was

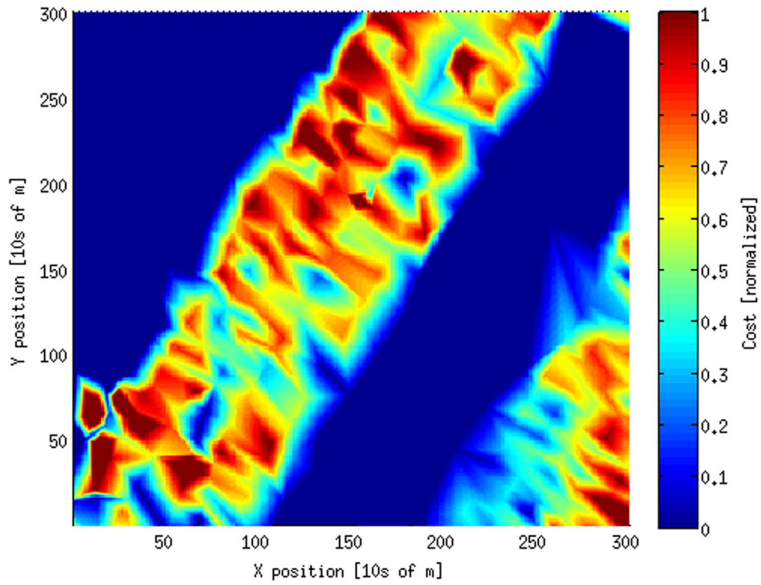


Fig. 7 Generated population cost map. Notice multiple areas in Manhattan have a population density of 1, and thus cannot be flown over or in by the UAS

given a higher cost to protect the UAS, encouraging a land-based sites to preserve the vehicle.

The terrain cost map is the weighted sum of these two terrain sub-cost maps, with weights of 0.2 given to the terrain slope cost map and 0.8 given to the terrain type cost map. The terrain cost map is shown in Fig. 9.

6 Planner

Using environment cost map C_E and the calculated path cost C_M, the planner must generate emergency landing paths that minimize risk to the environment as well as energy usage by minimizing the total cost C provided in Eq. 4.

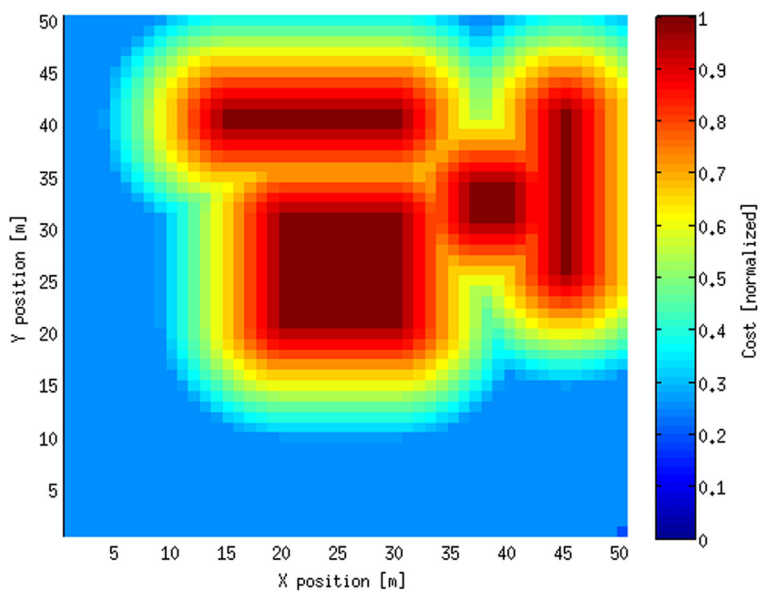


Fig. 8 Cost map with repulsion fields for four distinct buildings. Buildings are represented with a cost of 1.0 (*red*), with decreasing cost as a function of distance from all buildings decreasing to a baseline cost of 0.2 to allow for optimization on path length

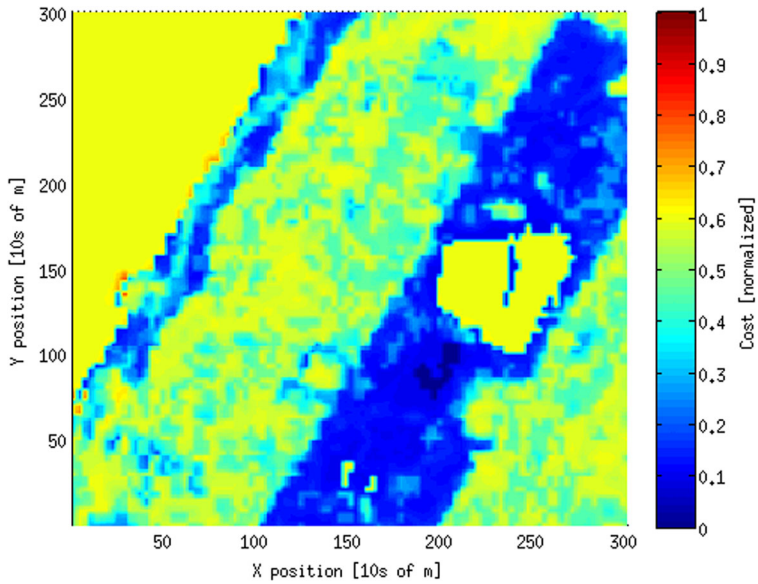


Fig. 9 Terrain cost map prioritizes areas that are flat, undeveloped, and not water, such as near the Hudson riverbank and Central Park

6.1 Planning Algorithm

A traditional A* search algorithm [16] is used to find the lowest cost emergency landing paths. The A* algorithm is an extension of Dijkstra's search algorithm that uses a heuristic to reduce the search space and provides an optimal path between start and end nodes given an admissible heuristic. The heuristic function is admissible if it never overestimates the remaining path cost during the search. The heuristic used in our A* implementation estimates cost to goal locations based on a calculated minimum cost per cell and estimated number of cells between the currently considered cell c and the closest potential landing location. A list of potential landing sites is generated by pre-processing the environment cost map. Candidate sites have environment cost less than a specified value C_{goal}. Also added to the list of candidate landing sites is a list of latitude/longitude locations of rooftops suitable for landing.[2] The minimum cost per cell is C_{goal}; all cells traveled by the UAS must cost more that C_{goal}, or the UAS would consider them a landing location. This heuristic is theoretically admissible in all situations, and therefore allows the emergency path planner to provide the lowest cost path.

[2]Presence of a rooftop site in our map implies the building owner has authorized rooftop emergency landings.

The single-layer A* planner defined in this work is well suited to multirotor flight planning because range is limited and there are no significant constraints on translational motion. Because the movement cost described in Section 5.2 appropriately penalizes acceleration, the A* approach will produce a feasible solution without incurring the complexity and convergence issues associated with optimal control [14] over a full dynamics model. Although not needed for this work, a path smoothing algorithm could be applied to the segmented A* solution to produce a trajectory with smooth accelerations.

Flight plan range and segment length together define search-space complexity because A* search-space depth is determined by total number of flight plan segments. For long-range plans, hierarchical search-space construction and definition of an optimal cell decomposition map will be essential to manage complexity. [26] However, the energy-constrained multicopter considered in this work seeks a nearby landing site which in turn supports a manageable A* search-space size.

Navigation in urban canyons, such those modeled in our New York City case study, may occur in a GPS-denied environment. Our planning algorithm assumes a sensor suite that accounts for loss of GPS in a manner that can provide robust navigation within and above the urban canyon environment. A recent survey of available urban navigation techniques [20] confirms

that it is reasonable to assume a sensor suite that is reliable in GPS-denied environments. In particular, statistics from Table A1 of Ref. [20] demonstrate that systems based on a combination of vision and IMU can offer comparable performance to GPS-based systems, particularly in a small UAS sufficiently robust to safely fly in and over densely populated areas such as Manhattan.

A* search begins from the current quadrotor state and terminates when it finds a path to an emergency landing location. In order to satisfy the energy constraint, the search space is bounded by an overestimate of the remaining flight range of the UAS and the generated path is checked afterward to ensure it can be flown with the energy remaining. Maintaining separate path costs on a per cell basis during the search requires as much memory as holding the environment cost map; it is much more memory efficient and only slightly less CPU efficient to check path cost after a path is found rather than during the search. Environment constraints are enforced during the search; if a cell or node in the environment cost map has a cost of 1, it is pruned from the search-space.

The planner has been implemented in C++. Since the environment cost map is discretized, cells are referenced by integer in a linear array. Integer math is both fast and precise, and this allows the storing of parents, necessary for path extraction after a landing location has been determined. The planner stores the environment cost map in an array of floating point numbers. In the interest of speed at the expense of memory, a pre-allocated array of integers with the same number of elements as the environment cost map is used to store the parents. A priority queue, initially with $\frac{1}{8}$ the number of elements as the environment cost map, is used to store the costs of visited cells. The pre-allocation of these arrays, while potentially requiring more memory than necessary, serves to eliminate or drastically reduce the delays of dynamically increasing the size of an array when more space is needed. As a large environment cost map may take 500MB of memory or more[3], reallocating arrays that may necessarily grow near this size takes a considerable amount of time.

[3]The 1501x1501x96 matrix from our case studies requires 825MB of memory

6.2 Computational Complexity

Due to offline pre-calculation of the entire three dimensional cost map, the only computational activity of significance at runtime is search. A* reduces the complexity of Dijkstra's, or uniform cost, search given an effective heuristic but adds the complexity introduced due to computing the heuristic function at each search node. The complexity of Dijkstra's algorithm is $O(n^2)$, where n is the number of cells in the three dimensional cost map. The complexity of A*, which maintains a priority queue of cells to be explored, is $O(O(h) \cdot n \log_2(n))$, where h is the heuristic function. The complexity of searching the space drops from $O(n^2)$ to $O(n \log(n))$ as Dijkstra's algorithm does not use a priority queue to sort potential nodes to visit while A* search does use a priority queue.

The heuristic function, as described above, calculates an estimate of the cost between the currently considered cell and the closest goal cell. The array of goals cells is iterated through linearly and the lowest estimate is returned, yielding a complexity of $O(g \cdot n \log_2(n))$, where g is the number of goal locations. For each cell explored, g goal locations are considered. Since the A* algorithm results in at most $O(n \log_2(n))$ cells being explored and the heuristic checks g goal locations, the final complexity is $O(g \cdot n \log_2(n))$.

Although the complexity of our A* implementation may not reduce the complexity of Dijkstra's if $g \cdot \log_2(n)$ approaches n, in practice the heuristic bounds the search space by up to nearly an order of magnitude, as shown in the results from the case study.

7 Case Study

For our case study, we used an area of Manhattan bounded by $(40.773^o\text{N}, 73.990^o\text{W})$ and $(40.800^o\text{N}, 73.954^o\text{W})$, which forms a 3 km by 3 km square. The cost map was formed in this region with 2m x 2m x 2m cubes, extending vertically 192m above ground level.

Manhattan is an ideal location for this case study. Relatively high average population density in comparison to the rest of the US means that population density can present a significant problem to path planning. The wealth of publicly available information

Table 1 Test matrix of latitude, longitude, and height positions with accompanying street locations and identifiers

ID	Latitude/Longitude	Street location	h_0
1	40.77754N, 73.97805W	Columbus between 72nd, 73rd	120
2	40.79913N, 73.96830W	Broadway and 103rd	40
3	40.78277N, 73.98234W	West End between 76th, 77th	80
4	40.77399N, 73.97860W	68th near Central Park	16
5	40.77399N, 73.97860W	68th near Central Park	40

about both the terrain and the structures for NYC creates a feature-rich cost map, providing a search that balances multiple safety factors.

7.1 Test Matrix

The results presented from our case study consist of 5 starting locations. Each starting location consists of an ID, a coordinate and height position, and the relative street location. Table 1 contains the starting locations, with initial height as h_0.

This test matrix represents various locations of the Upper West Side of Manhattan. Various rooftops were added as viable landing locations[4], based on a manual search using Google Earth™. The manual search determined the viability of a rooftop by characterizing the emptiness and slope of potential rooftops.

7.2 Results

The results of running simulations over the test matrix are presented in Table 2 below. For each location, isometric and top views showing cost and solution path is presented. Some locations showcase the differences in paths depending on starting height; others display flight in urban canyons, landing on rooftops, or flying over buildings for a more direct route. Table 2 presents results with accompanying time statistics: t_s, the plan creation time in seconds; t_x, the plan execution time (assuming 2m/s cruise speed) in seconds; and $t_s/(t_s + t_x)$, the ratio of plan creation time to total time.

The data presented in Table 2, interestingly, shows a lack of correlation between t_s and t_x. Although a longer plan should take more time to create, the size of the search space is the largest factor in plan creation time. Locations 4 and 5 both have a longer path length and, according to Table 3, both have longer travel distances as well. One would expect the search space for locations 4 and 5 to be larger than the search space for location 3. However, location 4 is in an urban canyon, and location 5 is just above an urban canyon; accordingly, the constraints of the buildings have eliminated a large fraction of the search space.

With this reasoning, location 1, with an altitude of 120m, should have a building-free and, therefore, large search space. This is the case; however, as noted in Table 3, location 1 has the smallest absolute distance from starting location to ending location and therefore a smaller search space than locations 3, 4, or 5. Additionally, location 2 has the smallest search space due to buildings and the direction that it needs to travel.

Note the path length in Table 3 is exactly twice the time t_s in Table 2. This is due to assuming a cruising speed of 2m/s with cells of 2m on each side; each cell requires one second to traverse, so the plan execution time is the same as the path length in cells, but half its physical length.

Another relationship apparent in this data is the nearly linear relationship between the percentage of space searched and t_s. This relationship is not perfectly represented in the time data due to the way

Table 2 Case study time statistics

ID	t_s	t_x	$t_s/(t_s + t_x)$
1	3.03	91	0.048128
2	1.34	84	0.024385
3	10.9	127	0.11744
4	4.10	136	0.037509
5	4.67	156	0.039409

Relatively low computation times allow for potential online implementation for small UAS. Low ratio of search time to execution time shows energy spent due to hovering during search is negligible in most cases. t_s is the average over 20 runs

[4]The authors again assume building owner permission has been granted to land on the indicated rooftops.

Table 3 Case study path statistics

ID	Path length (m)	% Space searched	% Energy used
1	182	0.68566	0.209
2	168	0.40920	0.192
3	254	0.68011	0.291
4	272	0.19371	0.312
5	312	0.19518	0.358

Energy use is linearly related to the length of the pathThe plan creation time is linearly related to the percentage of the search-space expanded

memory is allocated in the search algorithm. As more cells in the search space are searched, they are added to an array that expands lazily by doubling its size when required. As more cells are added, memory allocation requires more and more time.

A relatively high percentage of the space is searched for this example, indicating the heuristic is not efficiently directing the search to an optimal solution. In this case, all the starting locations have at least two disparate but acceptable landing locations that are considered by the heuristic, resulting in a greater portion of searched space than choosing a single landing location on which to concentrate. However, choosing a single landing location brings problems of its own; a potential location directly behind a building may be closer in terms of absolute distance and therefore preferred by the heuristic, but a location near one of the rear corners of the building could actually be closer in terms of the simulated UAS flight capability and should therefore not be given high heuristic cost.

7.2.1 Navigating Urban Canyons

When flying between parallel rows of buildings, the UAS only increases altitude if beneficial. When starting at location 4, the UAS plans a path to land in Central Park, shown in Fig. 10.

Since extra height is not necessary - the landing location is on the ground of Central Park - the planned path altitude stays below the building heights, not increasing altitude and wasting energy.

The path is much closer to the buildings on the south side of 68th street - this is to achieve lower path cost. The cost, as seen in Fig. 11 decreases linearly parallel to the street. Accordingly, the path runs parallel to the street as closely as possible, only straying over a building once as the repulsion cost in previous examples was too high for the UAV to fly over buildings.

Starting from the same location with an altitude of 40 meters results in a slightly different path, shown in Fig. 12.

Unlike the 16 m altitude case, starting at 40 m allows the UAS to fly directly into a lower cost area before directly descending to the chosen landing location. This case also illustrates the planner's preference to select straight paths as changing direction requires more energy than continuing straight. In addition, the planner also maintains the same altitude for over half of the path; the last third of the path is spent in a gradual descent until the vehicle is over the landing location.

At both starting altitudes, altitude is maintained until the UAS is over the landing location; this both prevents danger to any people in the vicinity of the UAS and represents the lowest energy path as well. It represents the lowest energy path due to the discretization of the three dimensional cost map; assuming a continuous cost map, the lowest energy path would of course be the straight line from the west edge of Central Park to the landing location. However, a "straight line" path through the discretized map results in a jagged path with many direction changes, resulting in a higher energy expenditure than the path returned by the planner.

7.2.2 Maintaining Safety with respect to People when Operating at High Altitude

When planning at high altitudes, the planner respects the population density information present in the cost map even though it may be operating over 100 meters of altitude. This can be seen in the following two case studies, both of which demonstrate a low energy condition occurring at altitude and landing on an appropriate rooftop nearby.

A low energy condition triggered at location 3 causes the planner to produce the flight plan shown in Figs. 13 and 14. This flight plan includes a descent to land on a rooftop.

Figure 13 shows the isometric view of the generated path, with a starting point near the tops of all the buildings around it and ending on a rooftop. While this figure shows the path, it does not adequately highlight

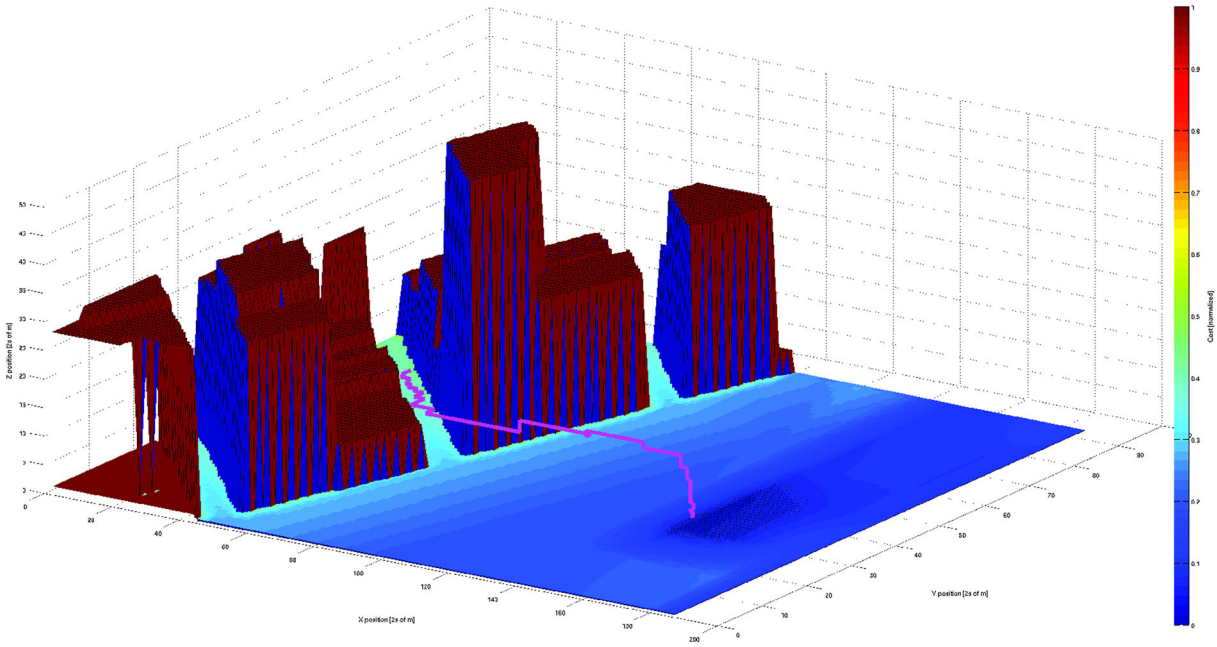


Fig. 10 Isometric view of the planned path through an urban canyon to land in Central Park, with an initial UAS altitude of 16 meters

another behavior of the planner, which is better shown in Fig. 14.

Figure 14 demonstrates the planner favors safety over energy efficiency. Since the cost map was gen-

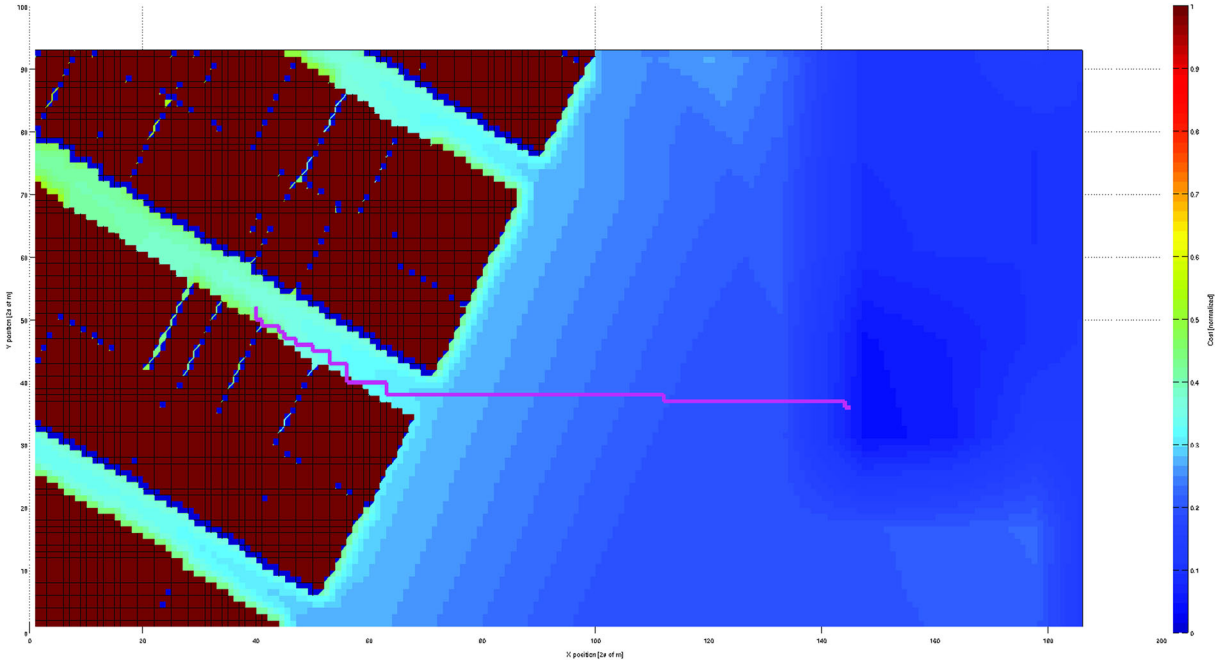


Fig. 11 Top view of the planned path through an urban canyon to land in Central Park, with an initial UAS altitude of 16 meters. The path is closer to the south side of 68th street than the north side

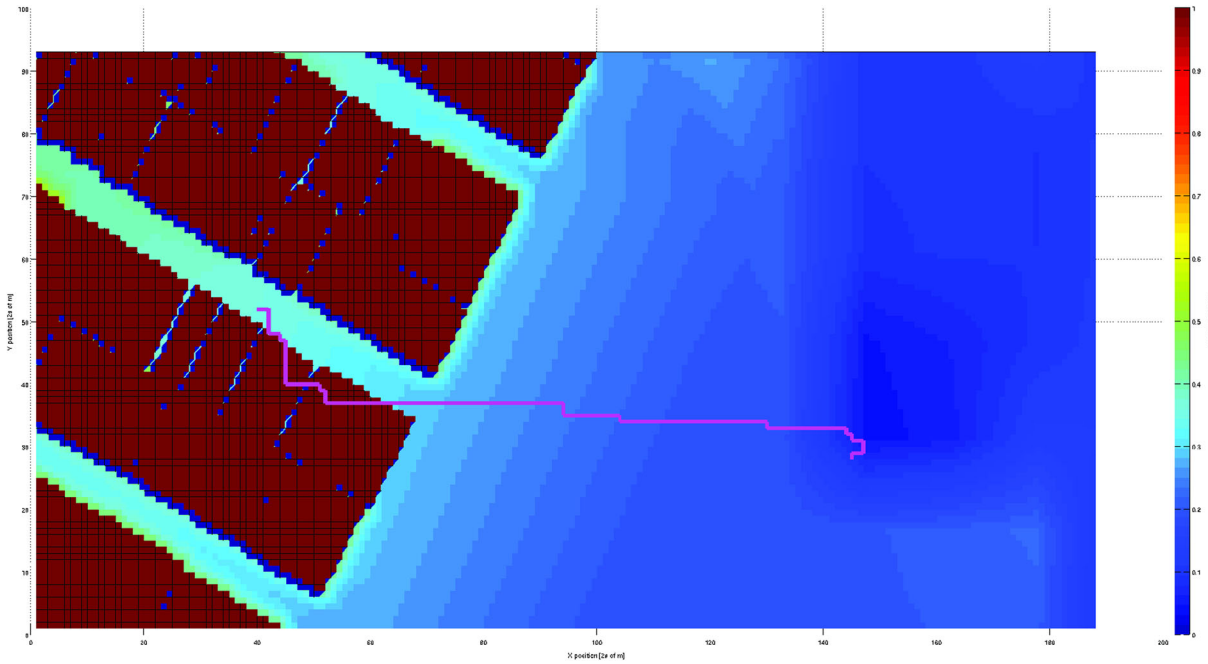


Fig. 12 Top view of the planned path above an urban canyon to land in Central Park, with an initial UAS altitude of 40 meters

erated with equal weighting to structural, population, and terrain costs, the maximum possible cost in an area without a structure is 0.66, represented by a yellow color. The landing location is in a large pool of yellow; the planner avoids much of the population and terrain cost, only planning a path over higher

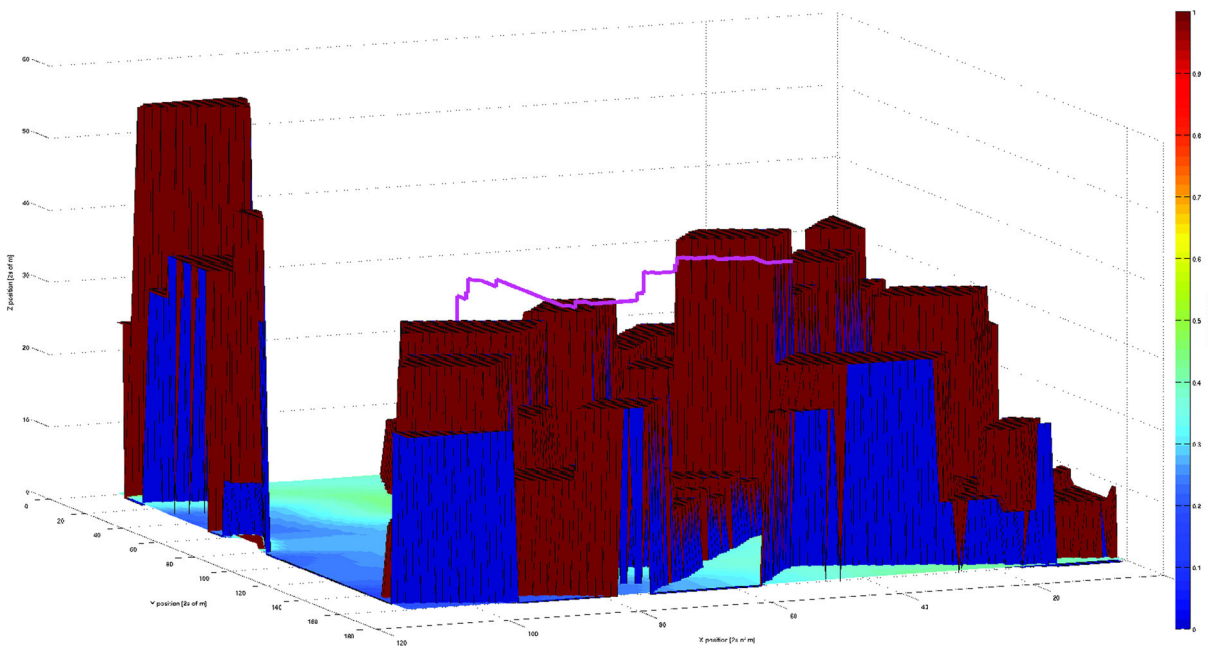


Fig. 13 Isometric view showing the generated plan from location 3 to the roof of a building

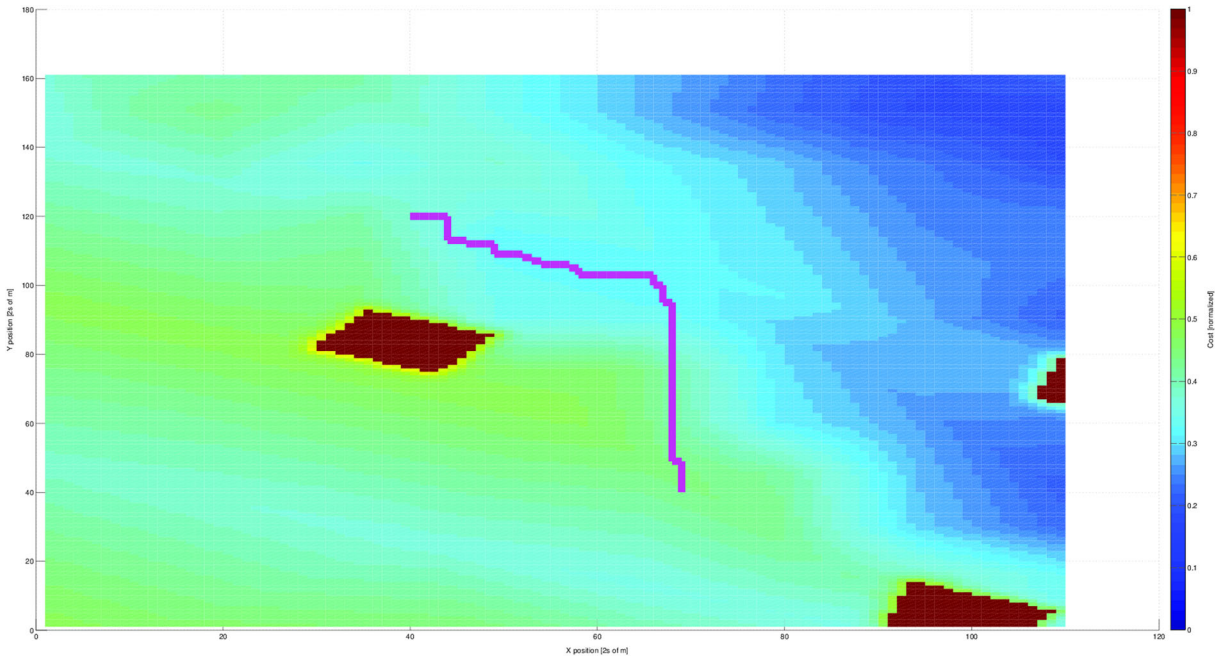


Fig. 14 Top view showing the generated plan from location 3 to a rooftop. Note that low cost is emphasized over energy efficiency

risk areas when it becomes too inefficient to avoid them.

The path from the planner after a low energy condition triggered at location 1 is shown in Figs. 15 and 16. Unlike the previously shown emergency landing path, this path features an ascent to land on a building with a height greater than UAS altitude when the low energy condition is triggered.

As can be seen in Fig. 15, the simulated UAS is in a large open space at an altitude of 120 meters when the low energy condition is triggered. The UAS slowly gains altitude as it approaches the building rooftop landing site. Due to the repulsors extending vertically from the building, the plan proceeds horizontally 6 meters above the roof; any people who happen to be on the roof are protected so long as the subsequent sensor-guided terminal approach to landing occurs properly.

As in Figs. 14, 16 demonstrates that the planner respects safety over energy efficiency until it is too inefficient to avoid these higher risk regions. Even though the entirety of the currently considered plan is above various buildings and never above a street, it is still considered safer by the authors - and by extension, the planner - to travel over a building with a lower population density than one with a higher population density.

7.2.3 Flight Over Buildings

While the planner makes efforts to take safety into account over pure energy efficiency, occasionally the safety added by a longer route is too inefficient to be mathematically justified in the planning algorithm. Consider the path shown in Figs. 17 and 18. The low energy condition is triggered when near a building at location 2.

Figure 17 depicts the simple flight plan: increase altitude, proceed over the building to directly above the landing location, and descend. Due to the repulsors extending vertically from the building the simulated UAS passes over, it remains 6 meters above the roof at all times.

The top view presented in Fig. 18 is especially interesting. In other case studies the planner has proceeded through low cost zones instead of preferring a direct path; here the planner goes through a slightly higher cost area at to conserve energy by preferring a straight path, not a longer path around the building it flies over.

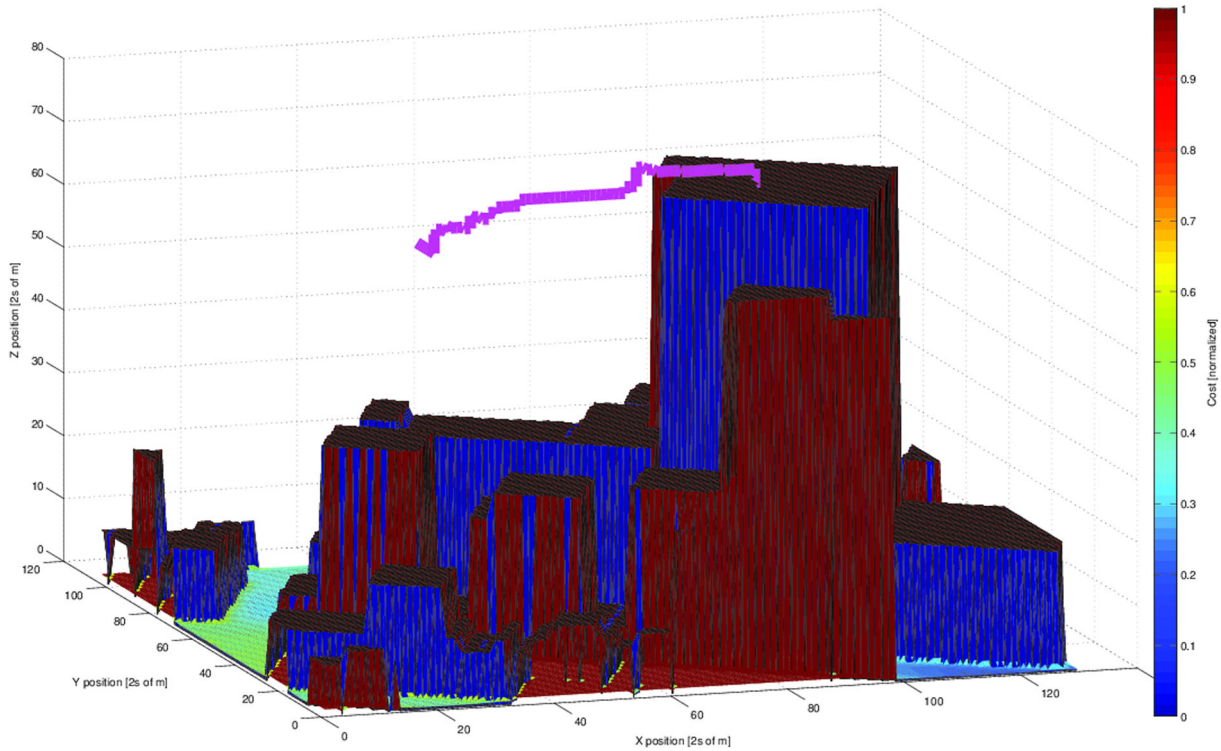


Fig. 15 Isometric view showing generated path. The UAS gains altitude to land on the roof of a building

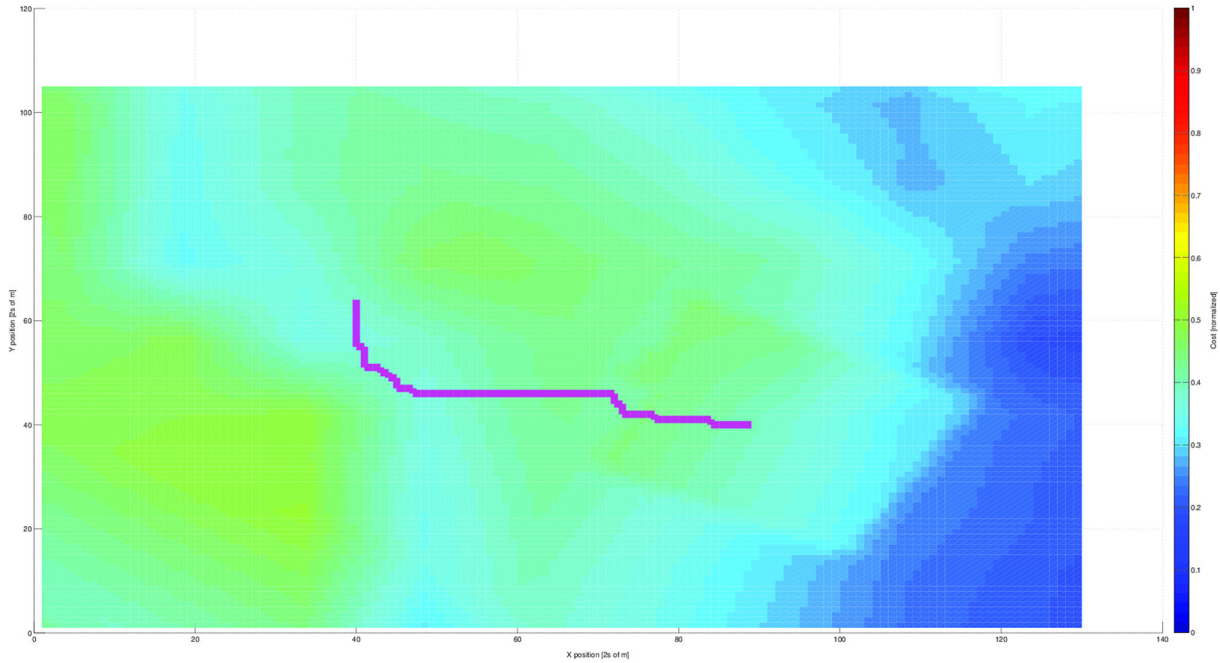


Fig. 16 Top view showing generated path. Even at an altitude greater than 100m, the UAS flies over the lowest-population areas possible

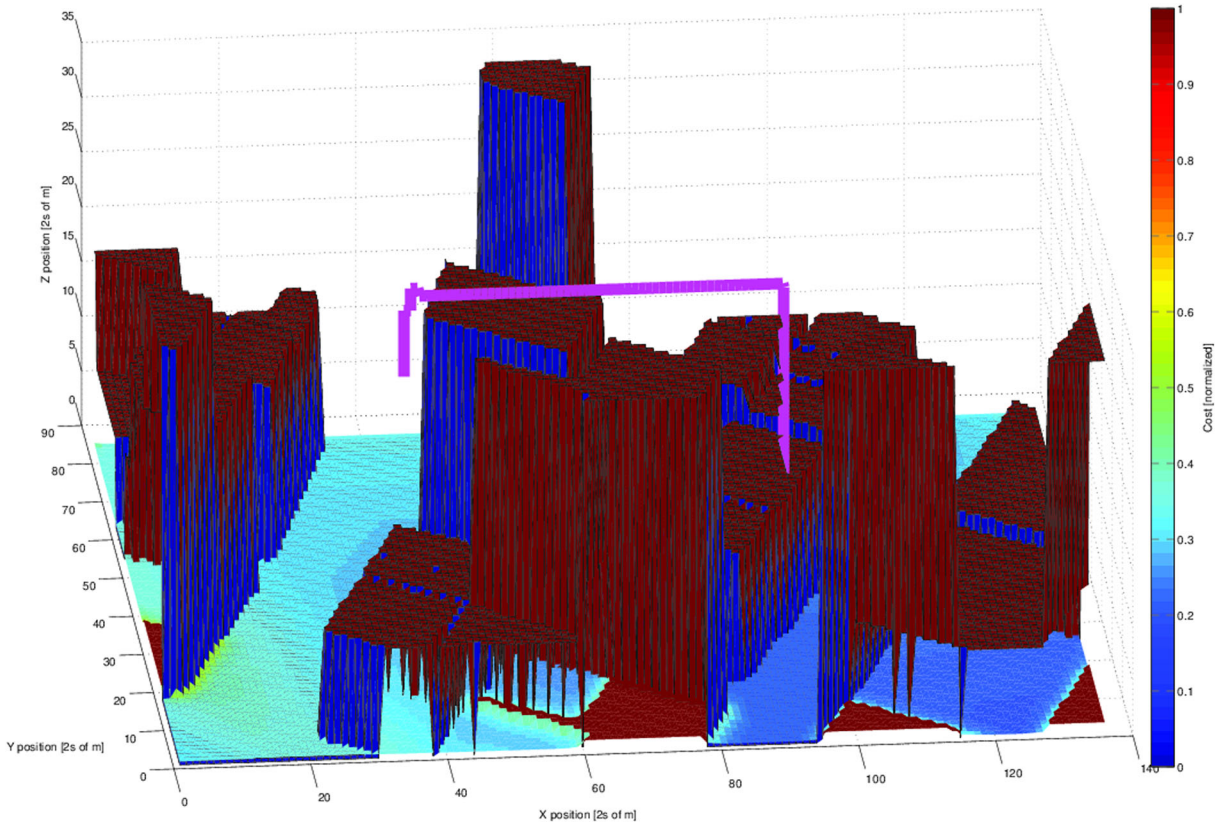
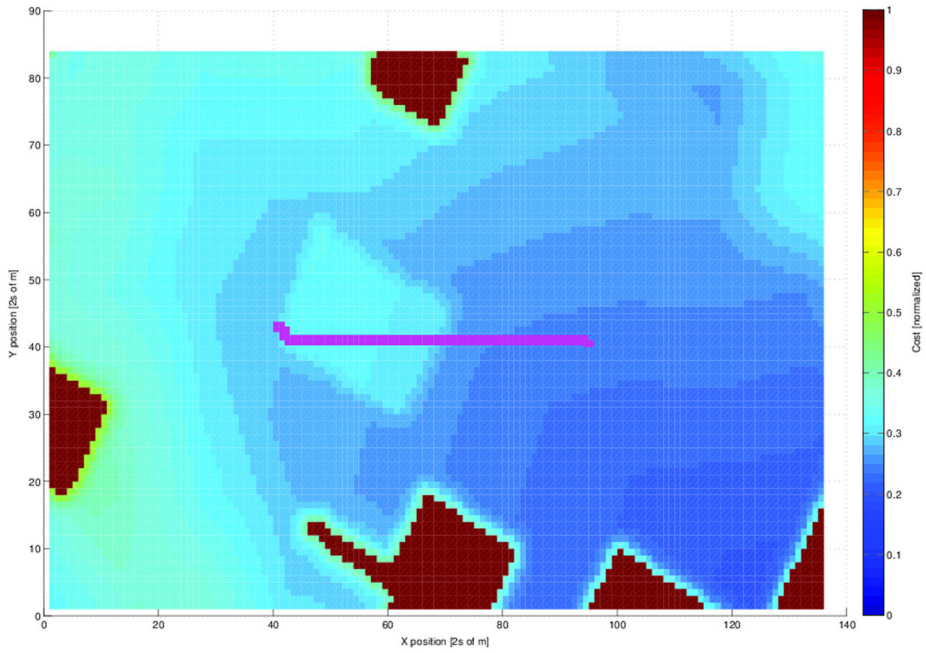


Fig. 17 Isometric view showing generated path. The simulated UAS gains altitude, flies over the building, and lands on a roof on the other side

Fig. 18 Top view showing generated path. While the planner considers safety, energy efficiency is not neglected, which is reflected in the straight path chosen



8 Conclusions and Future Work

This paper has presented a new method for emergency landing planning when a UAS unexpectedly encounters a low-energy situation. While a number of previous researchers have investigated emergency landing based on sensor feedback this work proposes a novel combination of sensor-based planning for terminal approach combined with a map-based planner to enable identification of a trajectory that extends beyond sensor range limited both by distance and environment occlusion. The primary contributions of this work are therefore in its incorporation of population, structure, and terrain data into the landing planner, devising a cost function and heuristic capable of balancing objectives during planning, and implementing the database and planner in a manner that can enable real-time execution. Simulations illustrate the efficacy of the map-based planner, and runtime statistics indicate the processed database and planning algorithm are feasible to store and execute onboard a small UAS. Database preprocessing transfers the bulk of the processing overhead out of the runtime map-based planner, greatly reducing the amount of required onboard computation.

A number of future improvements to the proposed path planner will produce a more robust system. Improvement to the UAS model will improve planning fidelity, and incorporating map data into alternative planning algorithms may further improve efficiency. Ultimately, the planner must handle cases where no viable landing site is within range of the vehicle, more comprehensively trading off risks associated with a non-ideal landing location with the probability of not having enough energy to reach the ground before losing power. New information could also be added to the cost function, e.g., expected traffic density information based on time-of-day and day-of-the-week, to prevent the UAS from designating the middle of a busy intersection or a crowded park as an acceptable landing zone.

Acknowledgments This research was supported in part by NASA contract NNX11AO78A. Special thanks to the MAAV team for sharing flight data from their quadrotor.

References

1. Artieda, J., Sebastian, J.M., Campoy, P., Correa, J.F., Mondragón, I.F., Martínez, C., Olivares, M.: Visual 3-d slam from uavs. J. Intell. Robot. Syst. **55**(4–5), 299–321 (2009)
2. Atkins, E.M.: Risk Identification and Management for Safe Uas Operation. In: IEEE International Symposium on Systems and Control in Aeronautics and Astronautics (ISSCAA) (2010)
3. Atkins, E.M., Portillo, I.A., Strube, M.J.: Emergency flight planning applied to total loss of thrust. J. Aircr. **43**(4), 1205–1216 (2006)
4. Bouktir, Y., Haddad, M., Chettibi, T.: Trajectory Planning for a Quadrotor Helicopter. In: Mediterranean Conference On Control & Automation, 2008. MED'08. IEEE (2008)
5. Bureau, U.S.C.: 2010 Census summary file 1 United States United States census bureau (2012)
6. Choudhury, S., Arora, S., Student, M., Scherer, S.: The Planner Ensemble and Trajectory Executive: a High Performance Motion Planning System with Guaranteed Safety. In: AHS Forum 70. AHS (2014)
7. Choudhury, S., Scherer, S., Singh, S.: Rrt*-Ar: Sampling-Based Alternate Routes Planning with Applications to Autonomous Emergency Landing of a Helicopter. In: 2013 IEEE International Conference On Robotics and Automation (ICRA), pp. 3947–3952. IEEE (2013)
8. Collins, T., Collins, J., Ryan, C.: Occupancy Grid Mapping: an Empirical Evaluation. In: Mediterranean Conference On Control & Automation, 2007. MED'07, pp. 1–6. IEEE (2007)
9. Di Donato, P., Atkins, E.: Exploring Non-Aviation Information Sources for Aircraft Emergency Landing Planning. In: AIAA Scitech Forum and Exposition (2016)
10. Environmental Systems Research, Inc., http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf: ESRI Shapefile Technical Description
11. Fitzgerald, D., Walker, R., Campbell, D.: Classification of Candidate Landing Sites for Uav Forced Landings. In: AIAA Guidance, Navigation, and Control Conference and Exhibit (2005)
12. Hayhurst, K., Maddalon, J., Miner, P., DeWalt, M., McCormick, F.: Unmanned Aircraft Hazards and Their Implications for Regulation. In: IEEE/AIAA 25Th Digital Avionics Systems Conference (2006)
13. Karaman, S., Walter, M.R., Perez, A., Frazzoli, E., Teller, S.: Anytime Motion Planning Using the Rrt*. In: 2011 IEEE International Conference On Robotics and Automation (ICRA), pp. 1478–1483. IEEE (2011)
14. Kirk, D.E.: Optimal control theory: an introduction Courier Corporation (2012)
15. Meuleau, N., Plaunt, C., Smith, D.: Emergency Landing Planning for Damaged Aircraft. In: ICAPS Workshop on Planning and Scheduling Applications (2008)
16. Norvig, P., Russel, S.: Artificial intelligence: a modern approach englewood cliffs (2005)
17. Olson, I., Atkins, E.: Qualitative Failure Analysis for a Small Quadrotor Unmanned Aircraft System. In: Guidance, Navigation, and Control Conference (2013)
18. Richardson, T.S., Jones, C.G., Likhoded, A., Sparks, E., Jordan, A., Cowling, I., Willcox, S.: Automated vision-based recovery of a rotary wing unmanned aerial vehicle onto a moving platform. J. Field Robot. **30**(5), 667–684 (2013)

19. Ro, K., Oh, J.S., Dong, L.: Lessons Learned: Application of Small Uav for Urban Highway Traffic Monitoring. In: AIAA Aerospace Sciences Meeting (2007)
20. Rufa, J., Atkins, E.: Unmanned aircraft system navigation in the urban environment: A systems analysis. Journal of Aerospace Information Systems (2016). doi:10.2514/1.I010280 (available online)
21. Saripalli, S., Montgomery, J.F., Sukhatme, G.: Visually guided landing of an unmanned aerial vehicle. IEEE Trans. Robot. Autom. **19**(3), 371–380 (2003)
22. Sharp, C.S., Shakernia, O., Sastry, S.S.: A Vision System for Landing an Unmanned Aerial Vehicle. In: 2001 IEEE International Conference On Robotics and Automation, Proceedings 2001 ICRA, vol. 2, pp. 1720–1727. IEEE (2001)
23. Thurrowgood, S., Moore, R.J., Soccol, D., Knight, M., Srinivasan, M.V.: A biologically inspired, vision-based guidance system for automatic landing of a fixed-wing aircraft. J. Field Robot. **31**(4), 699–727 (2014)
24. Warren, M., Mejias, L., Yang, X., Arain, B., Gonzalez, F., Upcroft, B.: Enabling aircraft emergency landing using active visual site detection. Field Serv. Robot. **9**, 9–11 (2013)
25. Watts, A., Ambrosia, V., Hinkley, E.: Unmanned aircraft systems in remote sensing and scientific research: Classification and considerations of use. Remote Sens. **4**(6), 1671–1692 (2012)
26. Zhu, D., Latombe, J.C.: New heuristic algorithms for efficient hierarchical path planning. IEEE Trans. Robot. Autom. **7**(1), 9–20 (1991)

Alec Ten Harmsel was introduced to multi-copters by the Michigan Autonomous Aerial Vehicles team as an undergraduate at the University of Michigan. He started doing research on emergency path planning and landing shortly after joining the team and enjoys working on other planning-related problems, such as navigating with computer vision. After finishing undergraduate education, he plans on working for Goldman Sachs, programming low-latency networked applications.

Ella Atkins Dr. Ella Atkins is an Associate Professor in the Department of Aerospace Engineering at the University of Michigan, where she is director of the Autonomous Aerospace Systems (A2SYS) Lab. Dr. Atkins research focuses on task and motion planning, guidance, and control to support increasingly autonomous cyber-physical Aerospace systems with focus on small UAS (unmanned aircraft system) and aviation safety applications. Dr. Atkins is author of over 150 refereed journal and conference publications and has served long-term as an associate editor of the AIAA Journal of Aerospace Information Systems (JAIS). She has served on numerous review boards and panels, including the 2013 NRC committee to develop a research agenda for autonomy in civil aviation. Dr. Atkins is past-chair of the AIAA Intelligent Systems Technical Committee, AIAA Associate Fellow, IEEE senior member, small public airport owner/operator (Shamrock Field, Brooklyn, MI), and private pilot. She served on the National Academys Aeronautics and Space Engineering Board (ASEB) (2011-2016) and was a member of the IDA Defense Science Studies Group (2012-2013). She currently serves on the steering committee and as Graduate Program Chair to the new University of Michigan Robotics Program.

Isaac Olson Isaac is a recent graduate of the University of Michigan with a Master's degree in Aerospace Engineering. He was introduced to multirotor UAS by the Michigan Autonomous Aerial Vehicles team as an undergraduate and has been working in flight controls and emergency path planning since. He currently works at SkySpecs, an Ann Arbor startup working to bring UAS to the structural inspection industry in a safe and effective manner.